Abstract
Hyperopt-sklearn is a new software project that provides automatic algorithm configuration of the Scikit-learn machine learning library. Following Auto-Weka, we take the view that the choice of classifier and even the choice of preprocessing module can be taken together to represent a single large hyperparameter optimization problem. We use Hyperopt to define a search space that encompasses many standard components (e.g. SVM, RF, KNN, PCA, TFIDF) and common patterns of composing them together. We demonstrate, using search algorithms in Hyperopt and standard benchmarking data sets (MNIST, 20-Newsgroups, Convex Shapes), that searching this space is practical and effective. In particular, we improve on best-known scores for the model space for both MNIST and Convex Shapes.

Experiments
Three data sets were used to conduct experiments on the effectiveness of hyperopt-sklearn.

MNIST Digits: A set of 28x28 greyscale images of hand drawn digits (60,000 images in the training set, 10,000 in the test set)

20-Newsgroups: A corpus of newsgroup messages that can be classified into 20 different categories (11314 articles in training set, 7532 articles in test set, used all 20 categories)

Convex Shapes: binary classification task of distinguishing pictures of convex white-coloured regions in 32x32 black-and-white images (8,000 images in training set, 50,000 in test set)

Optimization runs were performed on both the entire search space as well as subspaces corresponding to specific classifier types. Most experiments were run for 300 function evaluations of the parameter space. We used three optimization algorithms available in Hyperopt: random search, annealing, and TPE. The performance of the model found from searching the entire space was not statistically inferior to the best model pulled from each classifier subspace; there was no penalty for keeping all options open during search.

Comparison to Previous Work

<table>
<thead>
<tr>
<th>Approach</th>
<th>Accuracy</th>
<th>Approach</th>
<th>F-Score</th>
<th>Approach</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committee of covnets</td>
<td>98.9%</td>
<td>CFC</td>
<td>0.928</td>
<td>Hyperopt-sklearn</td>
<td>88.7%</td>
</tr>
<tr>
<td>Hyperopt-sklearn</td>
<td>98.7%</td>
<td>Hyperopt-sklearn</td>
<td>0.856</td>
<td>Hphi-dinet</td>
<td>84.6%</td>
</tr>
<tr>
<td>NaiveBayes grid search</td>
<td>98.6%</td>
<td>SVMorch</td>
<td>0.846</td>
<td>Ovm-3</td>
<td>81.4%</td>
</tr>
<tr>
<td>Boosted trees</td>
<td>98.5%</td>
<td>LBSVM</td>
<td>0.843</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the 20 Newsgroups dataset, the score reported for hyperopt-sklearn is the weighted-average F1 score provided by sklearn. The other approaches shown here use the macro average F1 score.

Example Usage

```python
from hyperopt import hp, tpe, fmin
from sklearn.datasets import fetch_20newsgroups
from sklearn.svm import SVC

# Download the data and split into training and test sets
train = fetch_20newsgroups( subset='train' )
test = fetch_20newsgroups( subset='test' )
X_train = train.data
ty_train = train.target
X_test = test.data
y_test = test.target

# Configure the search space
estim = fmin( hp.hyperopt_estimator( classifier='clf', preprocessing='tfidf', algo=tpe.suggest, trial_timeout=100),
             [X_train, y_train],
             print( 'estim.best_model()' )
)

# Report the parameters used in the best model found by the fit method
print( estim.best_model() )

# Report the accuracy of the model on the test set
print( estim.score( X_test, y_test ) )
```